LA-UR-22-26727

Approved for public release; distribution is unlimited.

Title: LANL's Developments for ZFS for NVMe based Parallel File Systems

Author(s): Atkinson, Brian Wayne

Intended for: Sharing LANL's developments with ZFS and NVMe devices.

Issued: 2022-07-11

Los Alamos
NATIONAL LABORATORY

Los Alamos
NATIONAL LABORATORY

LANL's Developments for ZFS for NVMe Based Parallel File Systems

Brian Atkinson
HPC-DES Storage Team

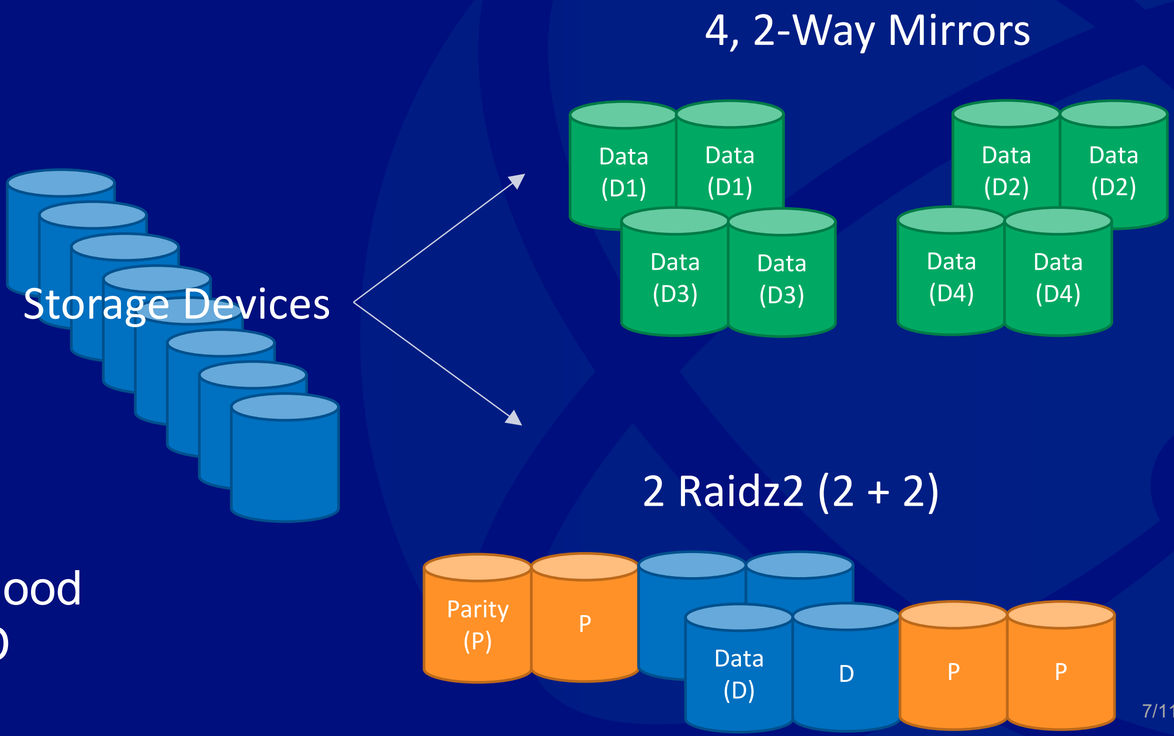
July 21, 2022

Agenda

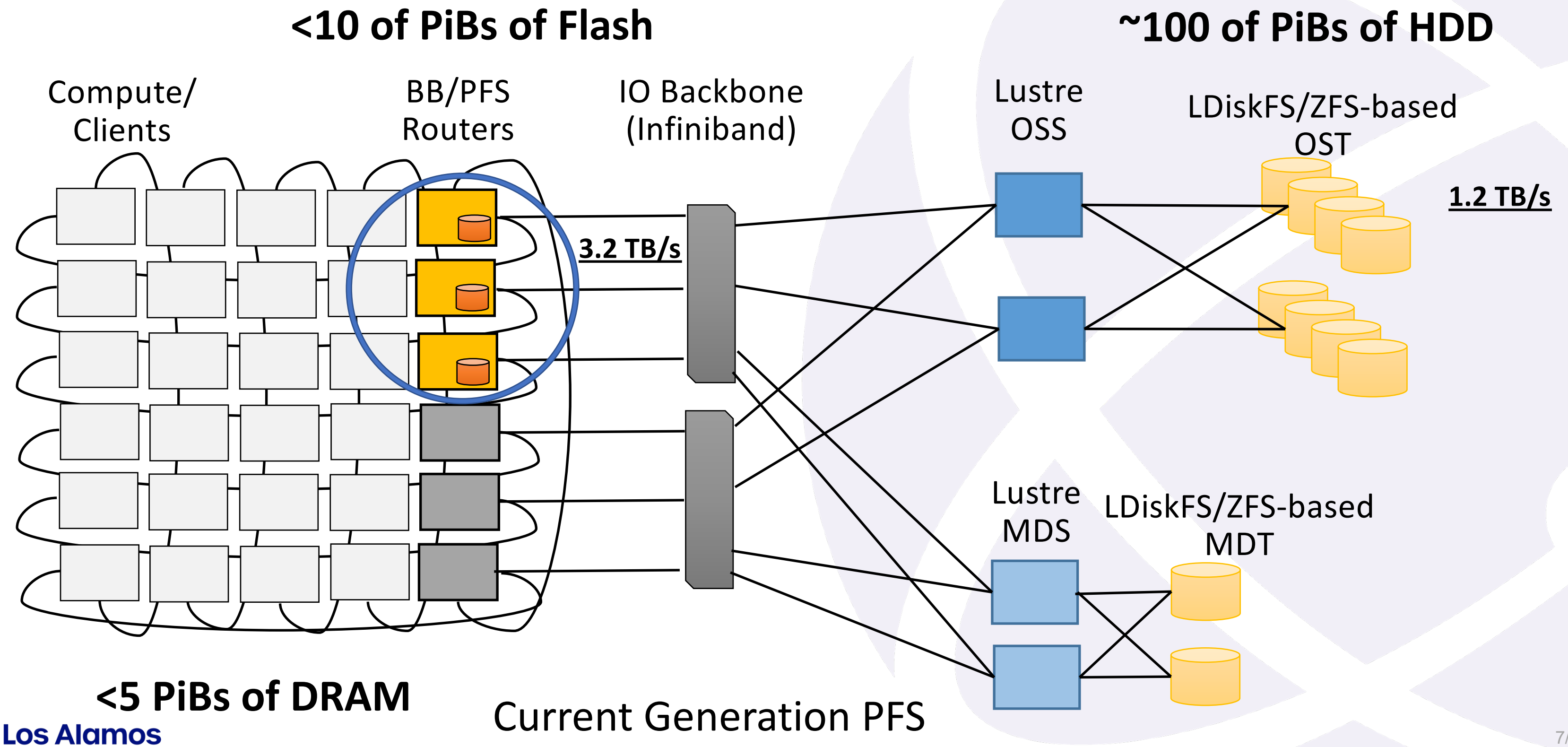
- Why is LANL invested in ZFS?
- Why are we focusing on NVMe device performance with ZFS?
- Two parallel projects for improving ZFS performance with NVMe devices
  - Adding O_DIRECT to ZFS
  - Incorporating computational storage devices into ZFS

Why Does LANL Care about ZFS?

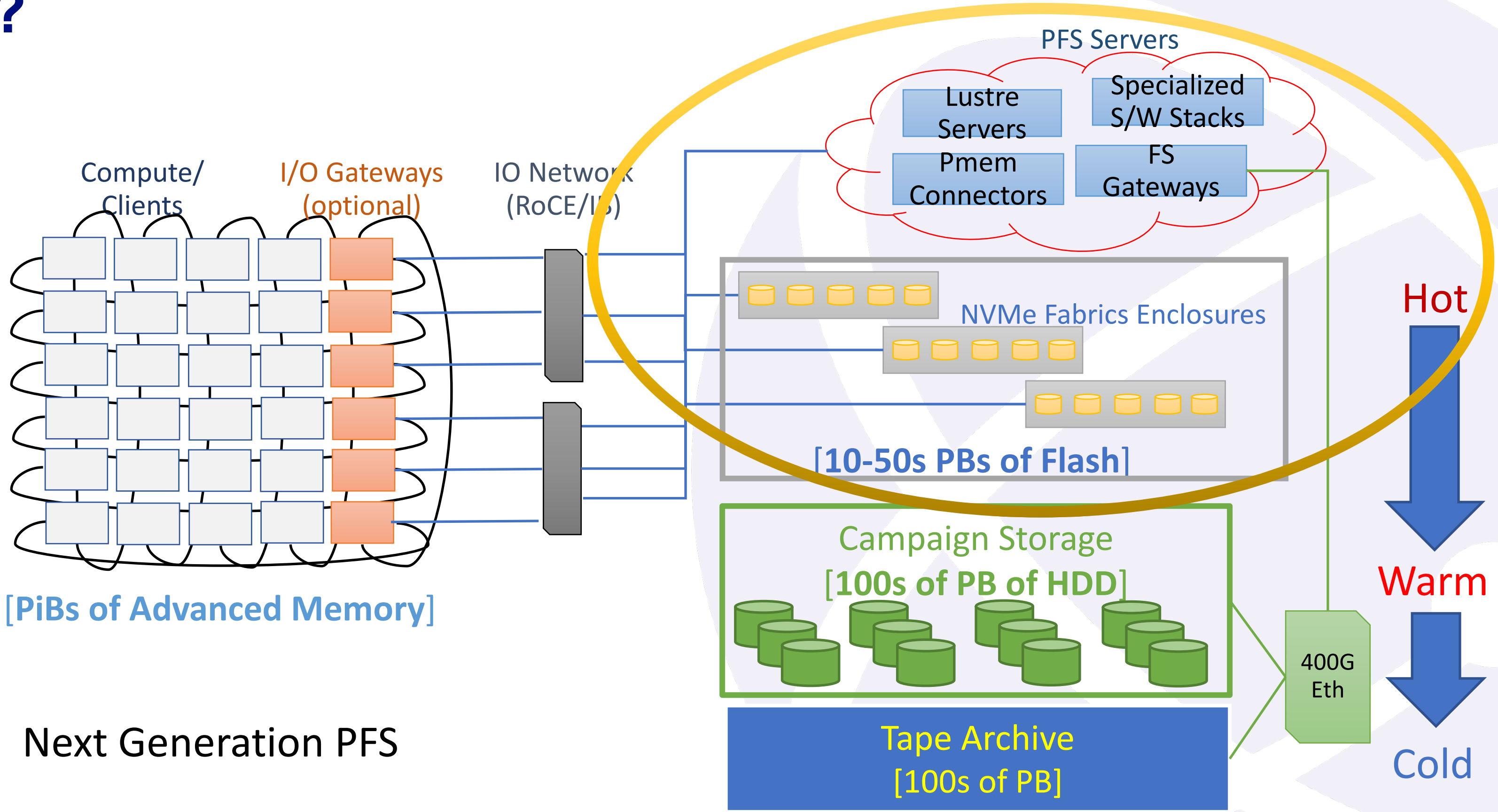
- **One of two available backing FS's for <u>Lustre</u>**
- Open sourced
- High integrity
  - Erasure coding (raidz)
  - Mirrors
  - Checksums
  - Snapshots
- Feature rich
  - Compression
  - Dedup
  - Encryption
- ZFS traditionally has good performance with HDD



Why are we focused on NVMe Device Performance with ZFS?



Why are we focused on NVMe Device Performance with ZFS?



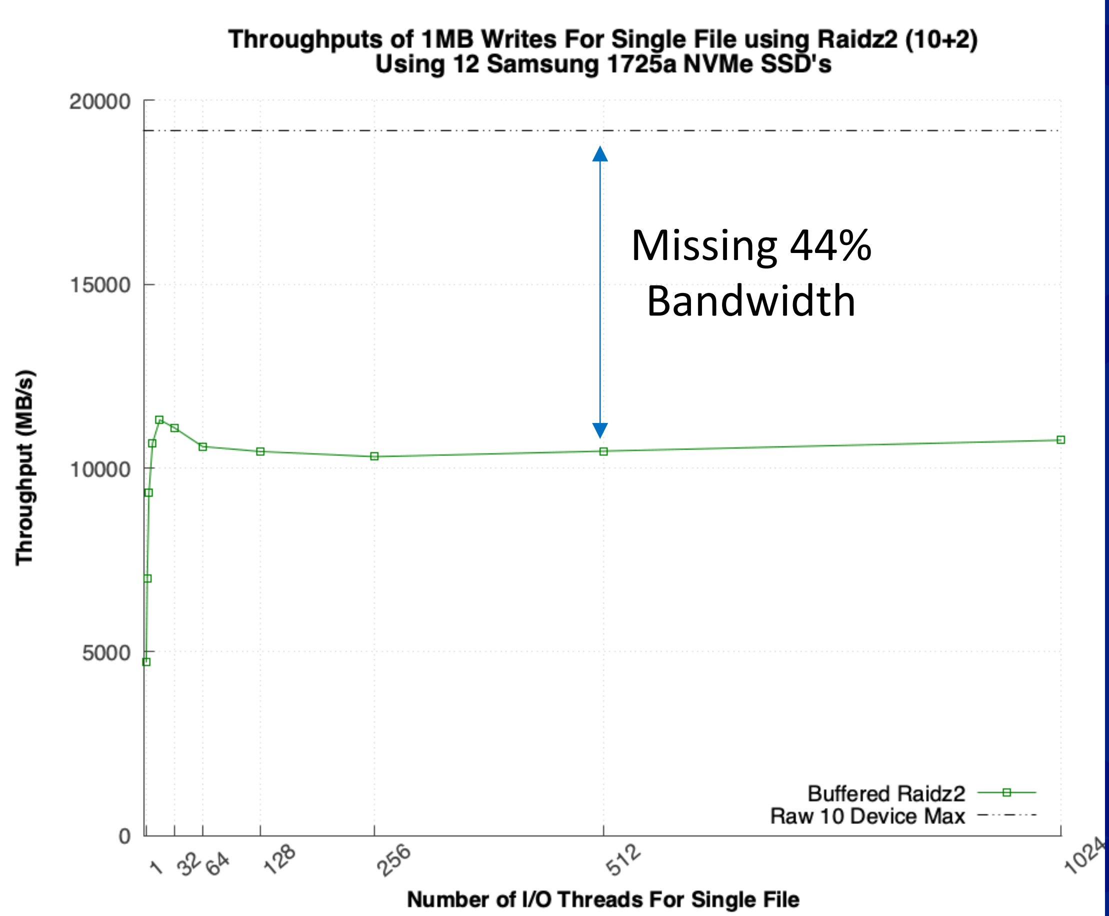
Next Generation PFS

Why are NVMe Devices performant?

- Solid state (Flash based)
- Connected through PCIe root complex
- High throughput (Gen4 2 GB/s per PCIe lane)
- Low latency (Millions of IOPS)
- Shrinking overall Lustre capacity
  - Need to capture all available bandwidth
  - Need to efficiently use storage capacity

NVMe Devices

Project 1: Addressing ZFS NVMe SSD Zpool Performance

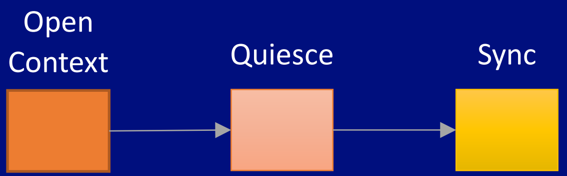


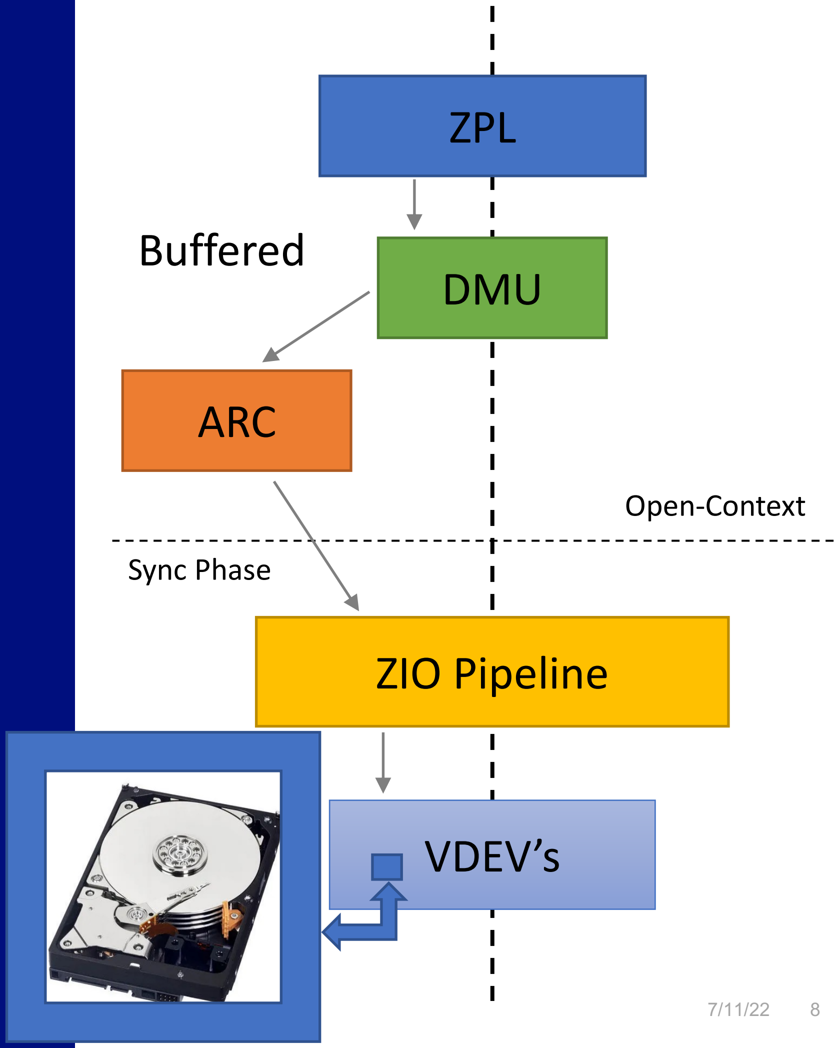
Los Alamos NATIONAL LABORATORY

How does ZFS Writes Data (Highlevel)

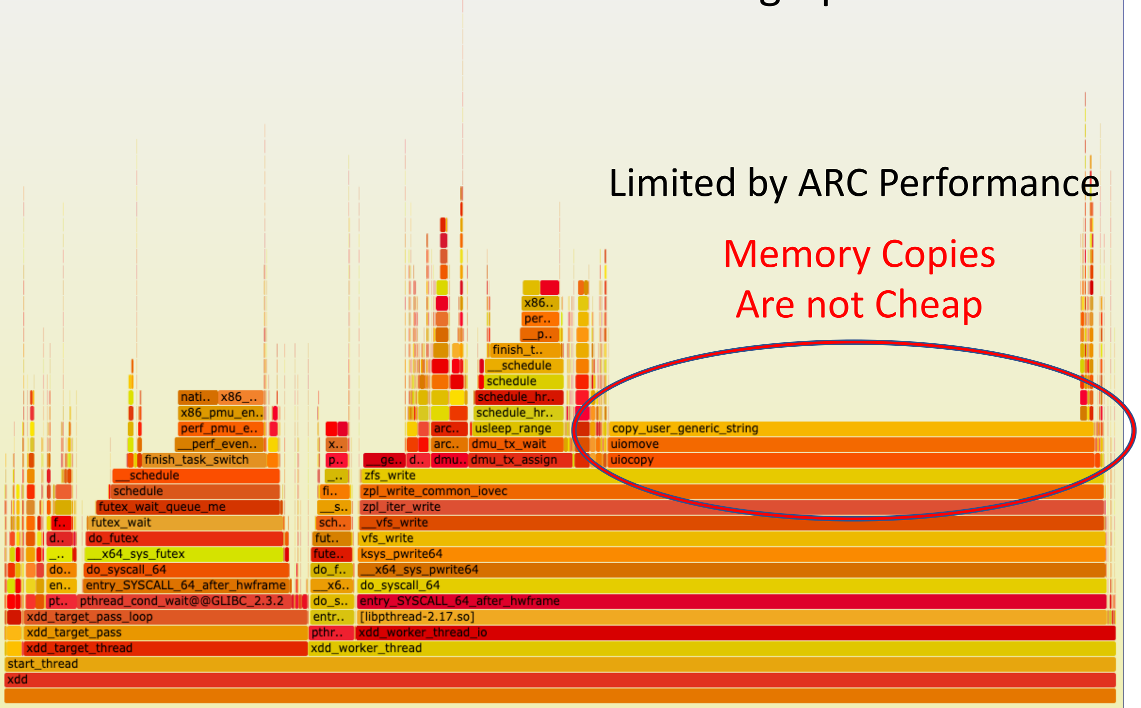
- Data written to ARC (system call returns)
- Write assigned to TXG in open-context
- Eventually written to disk(s) during sync phase of TXG

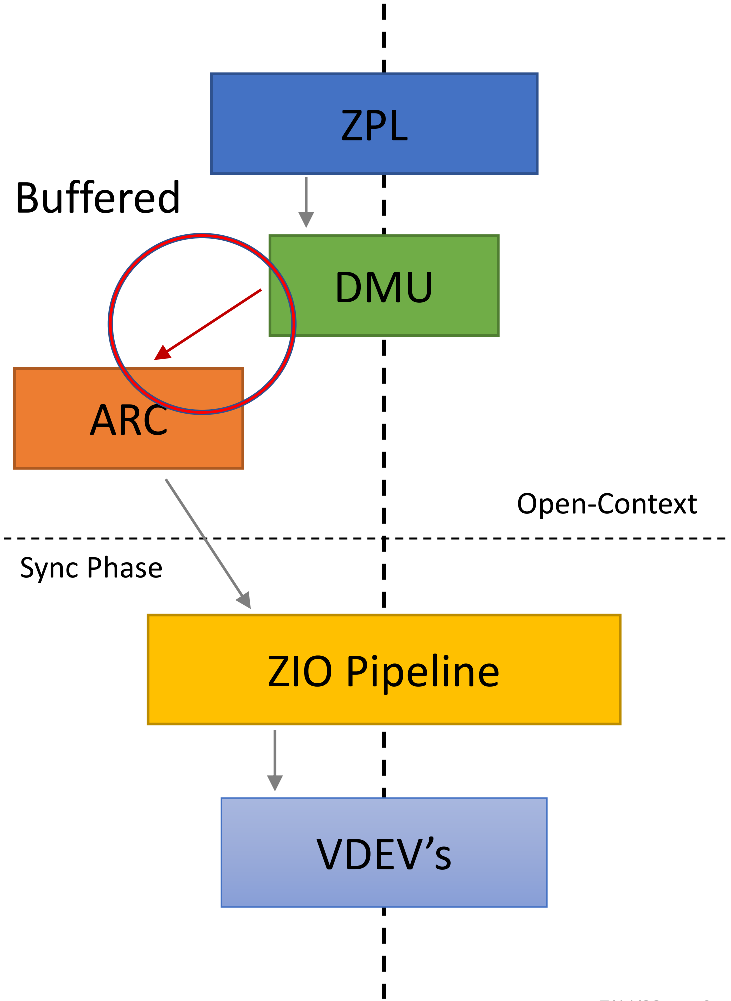
Transaction Group (TXG) Lifecycle





ZFS Buffered Write Flamegraph

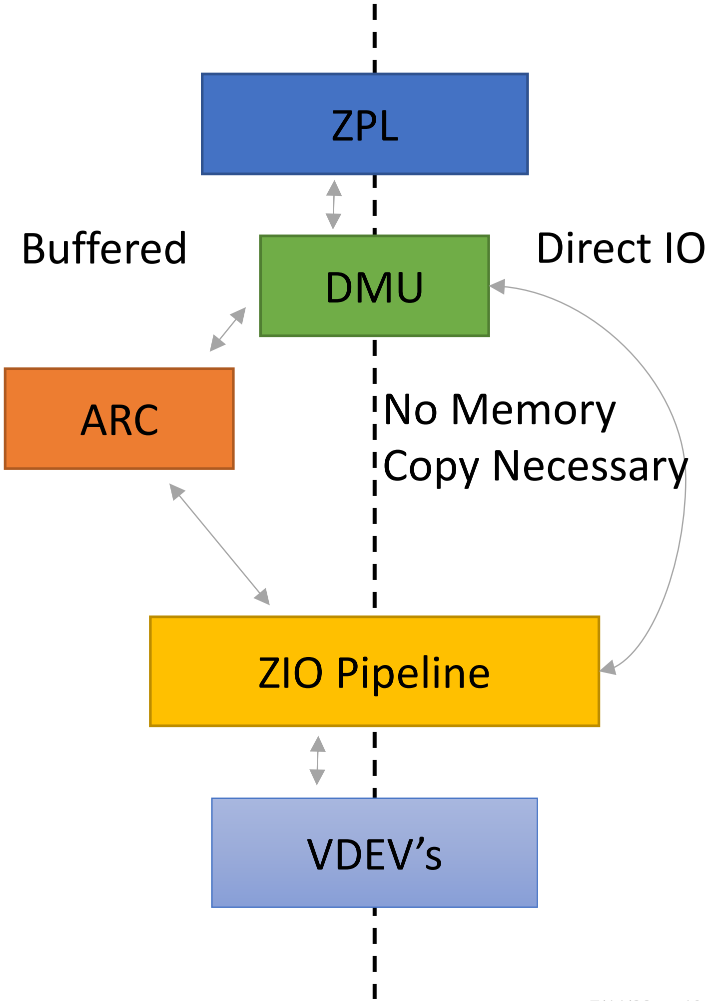




Simple Solution: Implement Direct IO into ZFS

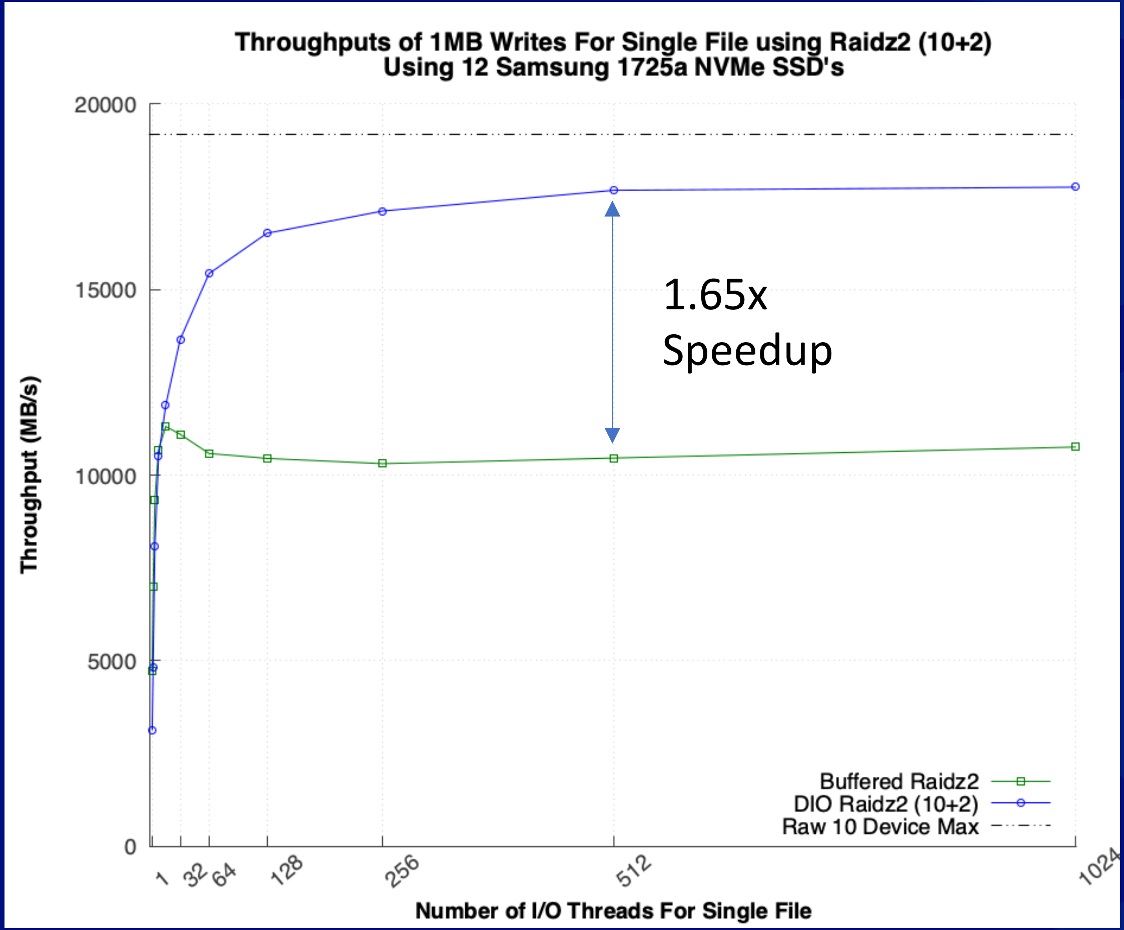
- What Exactly is Direct IO
  - Pass O_DIRECT flag in open() call
  - From the Linux man page for open():
    - Try to minimize cache effects of I/O to and from this file… File I/O is done directly to/from user-space buffers.

- Direct IO in ZFS
  - Currently ZFS silently ignores O_DIRECT
  - With update we Bypass the ARC
  - User pages are read/written from directly

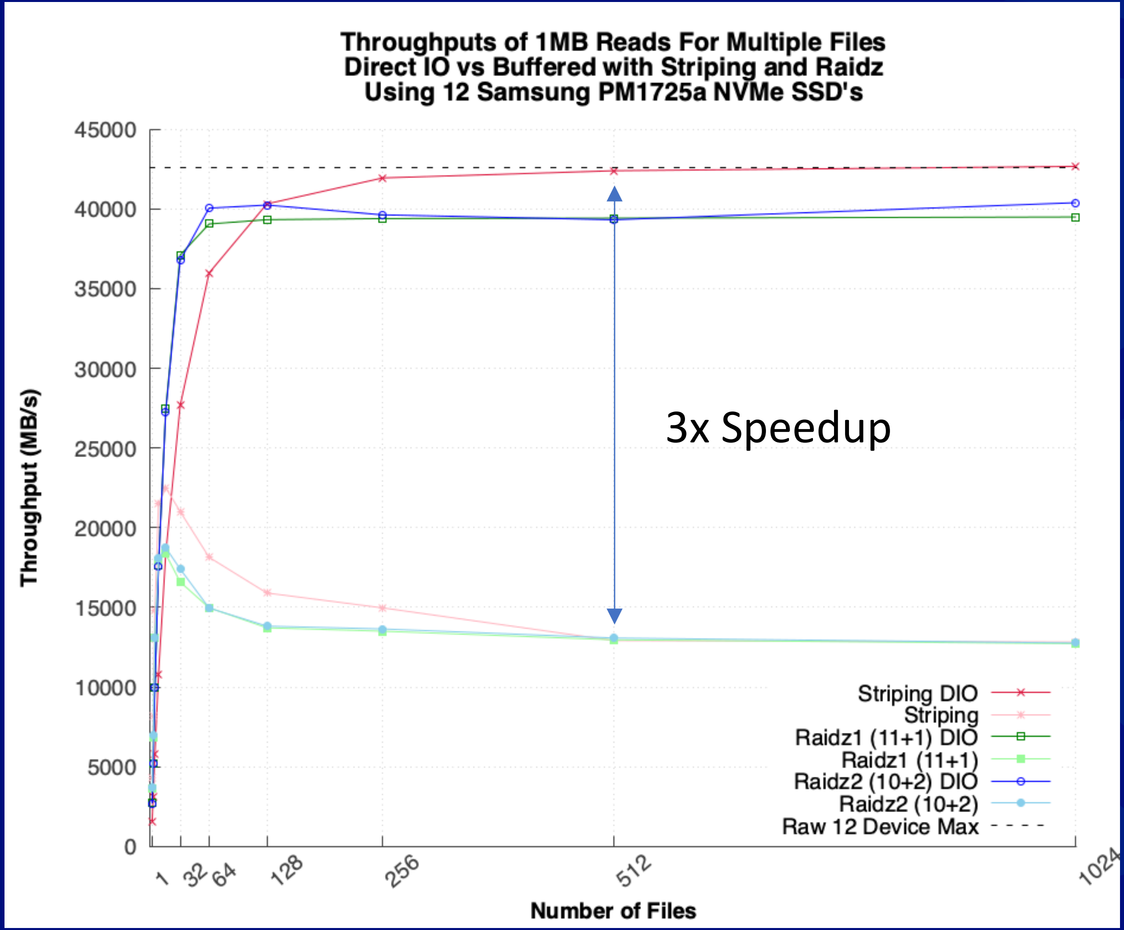


Los Alamos NATIONAL LABORATORY

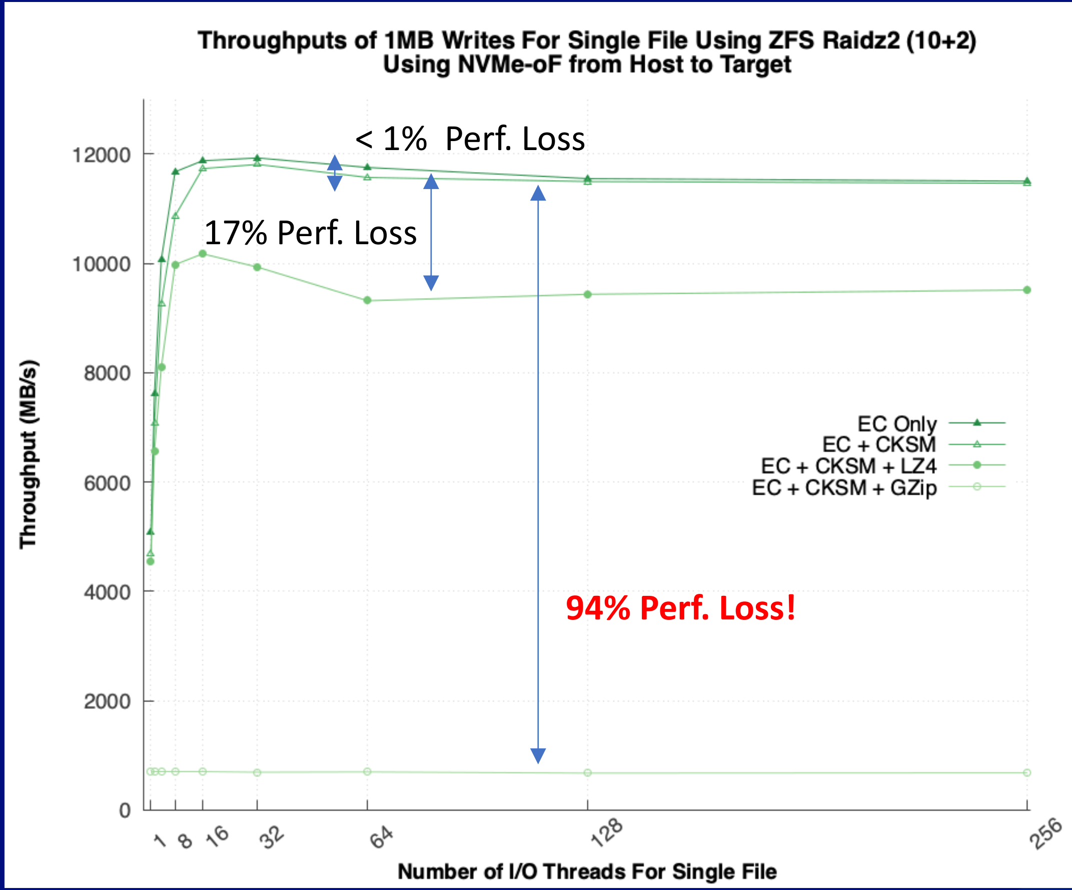
Seq. Write Performance Buffered vs Direct IO



Seq. Read Performance Buffered vs Direct IO

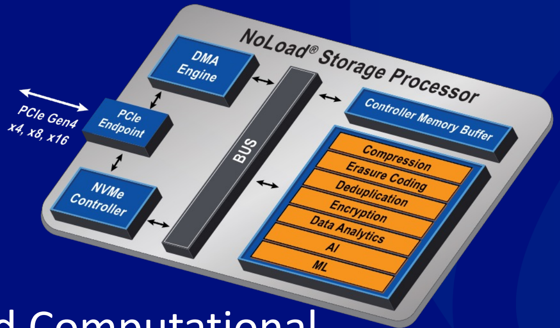


Project 2: Addressing ZFS CPU and Bandwidth Intensive Operations with NVMe SSD Zpools



What can do to improve performance?

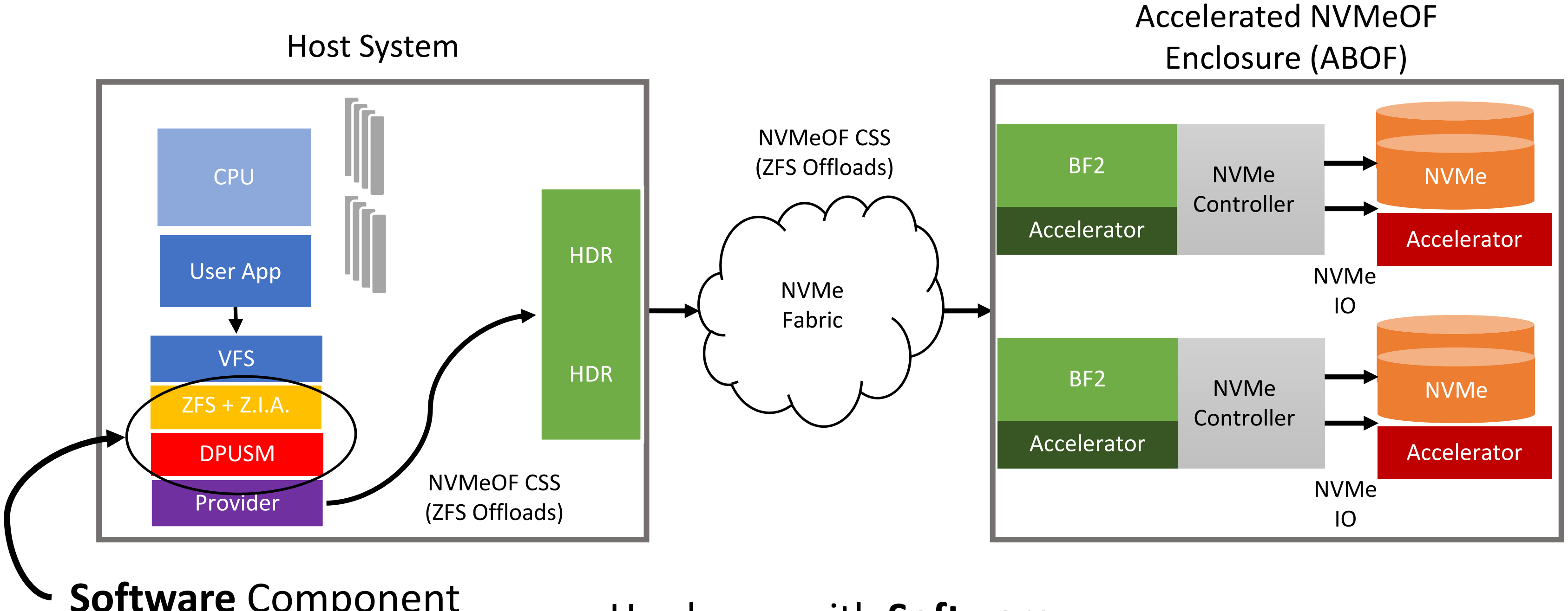
- Use computational storage devices (Accelerators)
- Computational storage devices
  - NVMe devices
  - Can offload CPU/Memory bandwidth intensive operations
  - Can be computational storage processor (FPGA) or even data processing unit (DPU)
- Transformations of ZFS data can leverage them



Eideticom Noload Computational Storage Processor (CSP)

NVIDIA BlueField2 DPU

One problem: How do you use these devices? Co-design

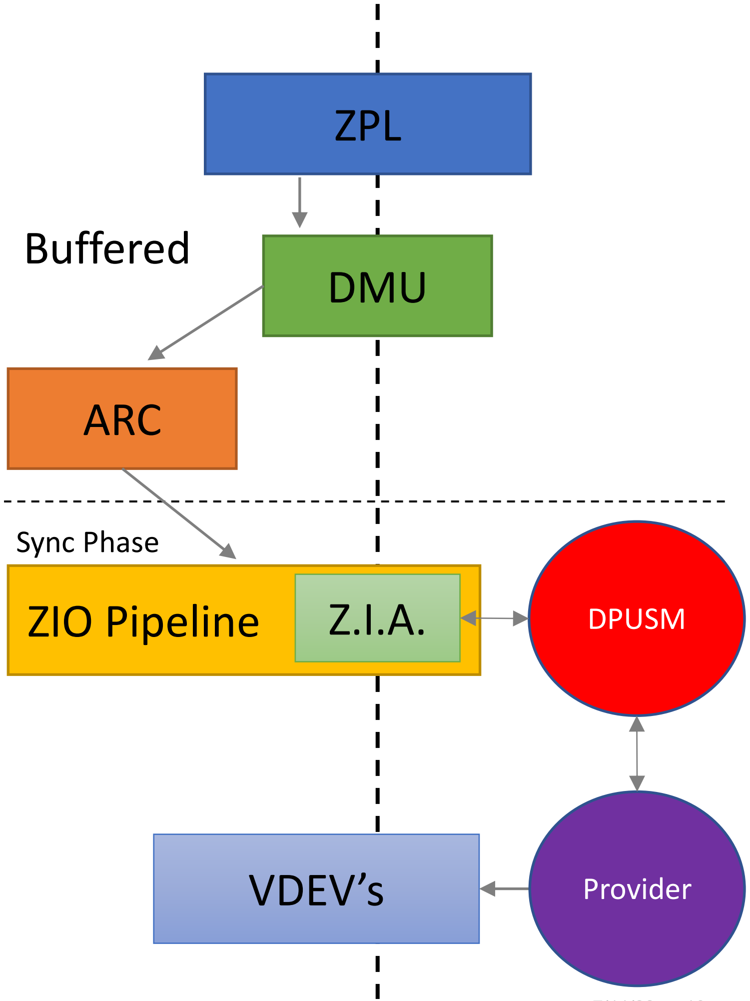


Software Component

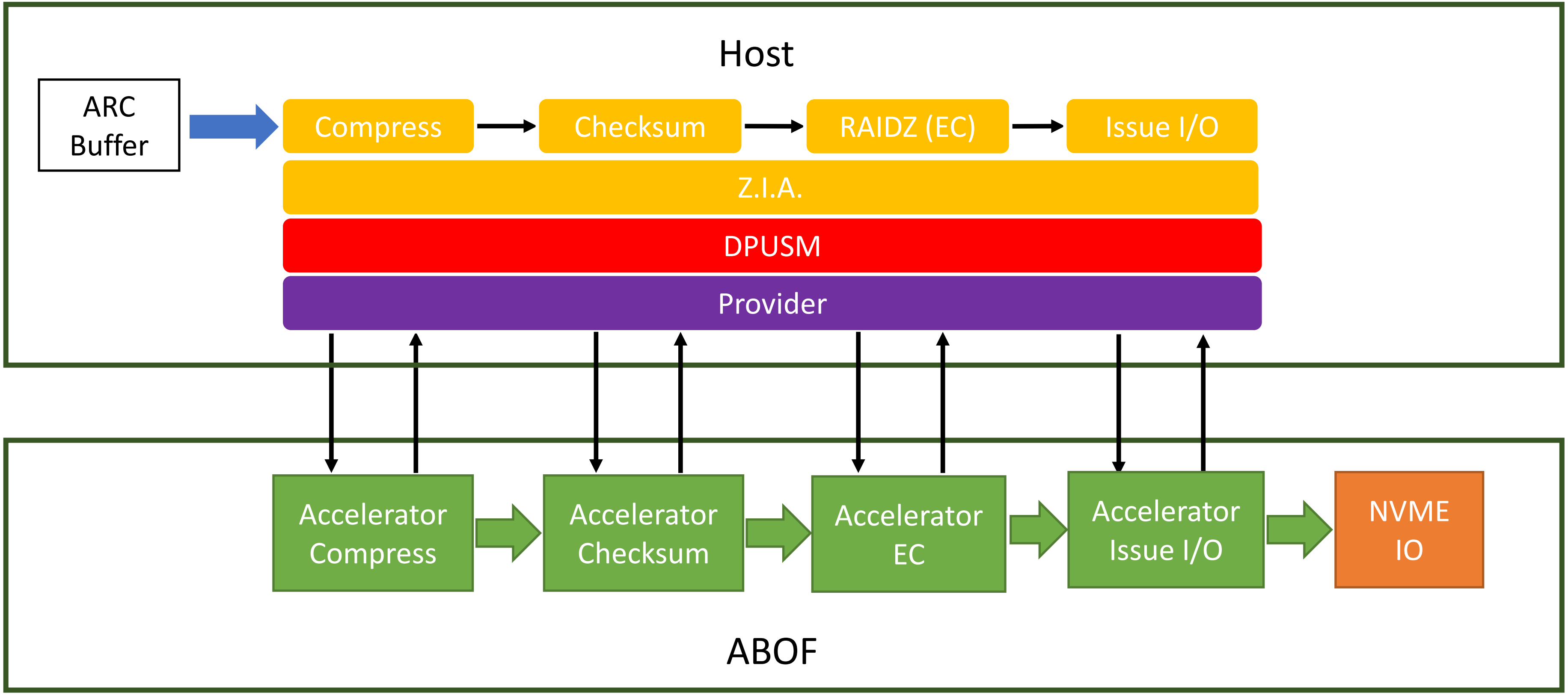
Hardware with **Software**

Software Layer Consists of 2 Components

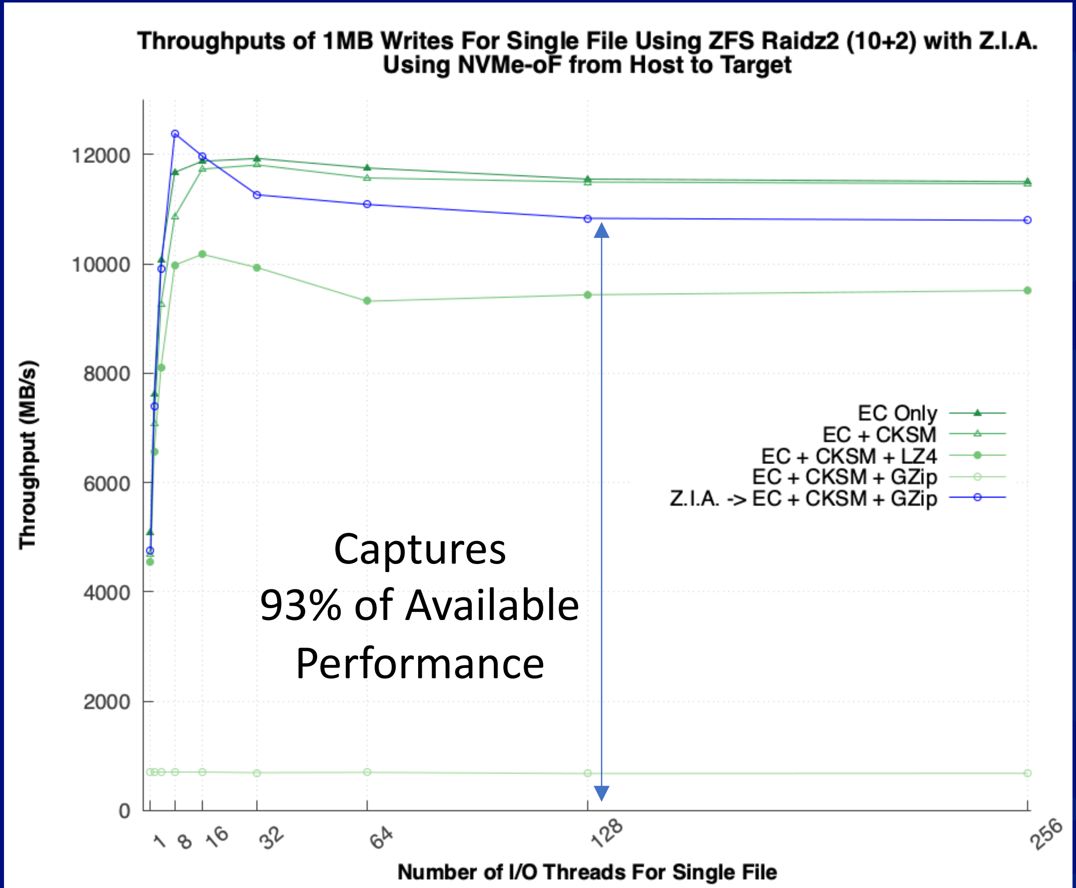
- DPUSM
  - Kernel module
  - Standardized API for leveraging computational storage devices
  - Can be expanded
  - Communicates with Providers (Accelerator specific code)
- ZFS Interface for Accelerators (Z.I.A.)
  - Bridge between ZFS and DPUSM
  - Allows ZFS to offload memory and CPU intensive operations
    - Compression, Checksum and EC



Software Layer Detailed View



ZFS Data Integrity and transformation Performance with Z.I.A.



Current Status of These Projects

- O_DIRECT is a PR to OpenZFS master
  - Should be merged soon
  - Work on Lustre patches to use O_DIRECT with ZFS
- Z.I.A.
  - Just opened a PR to OpenZFS master
- Plan to combine both projects into one to improve ZFS overall performance with NVMe devices

Thank You!

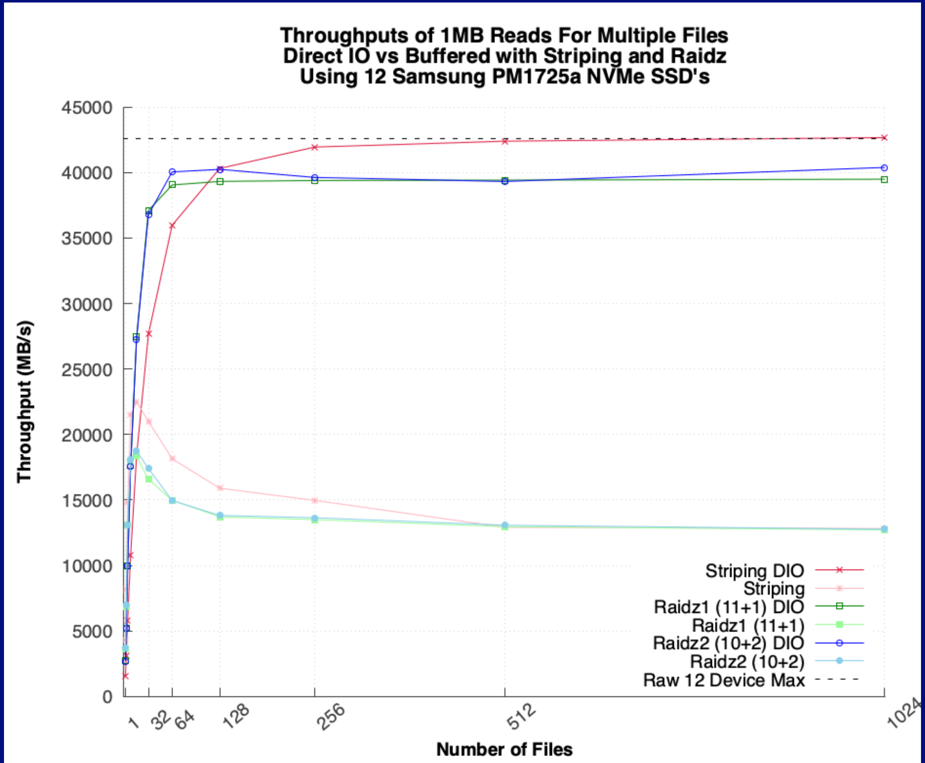
References

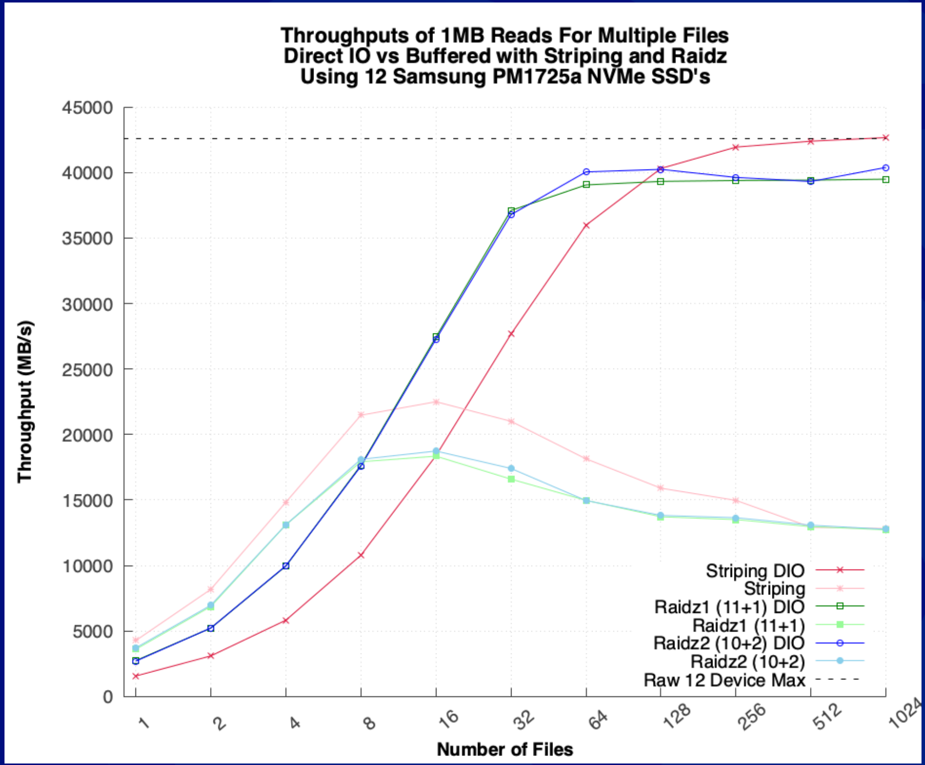
- Slide 4: Photo “NVMe Devices”, https://nvmexpress.org/portfolio-items/huawei-es3000-v5-series-nvme-ssd-storage-device
- Slide 6: Photo, “Hard disk internals”, https://www.indiamart.com/proddetail/hard-disk-internals-13413368373.html
- Slide 12: Photos, “Eideticom Noload Computational Storage Processor (CSP)”, https://www.eideticom.com, https://www.eideticom.com/products.html
- Slide 12: Photo, “NVIDIA BlueField2 DPU”, https://www.nvidia.com/en-in/networking/products/data-processing-unit

Addendum Slides

ZFS Buffered Write Flamegraph

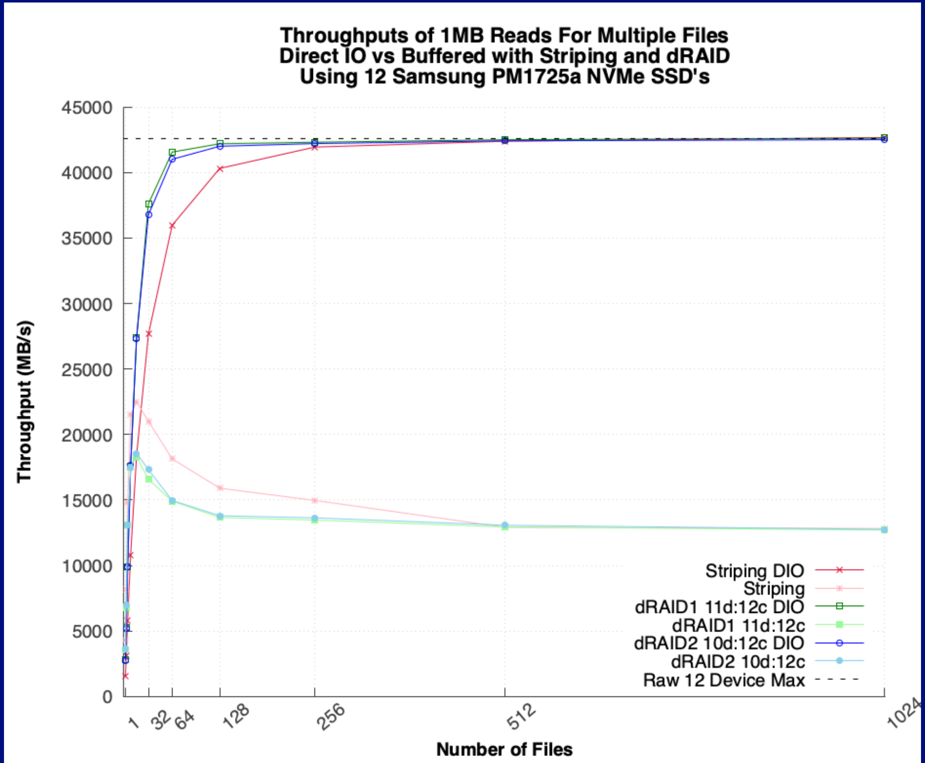
Seq. Read Performance Results: ZFS NVMe Zpools Raidz

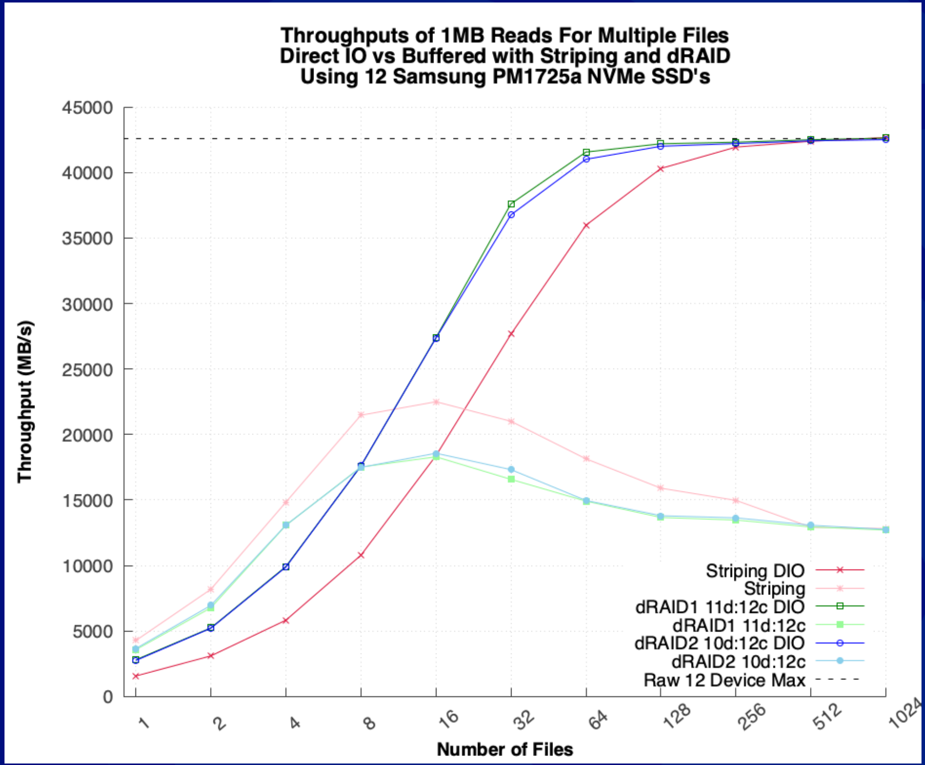




Log Scale

Seq. Read Performance Results: ZFS NVMe Zpools dRAID

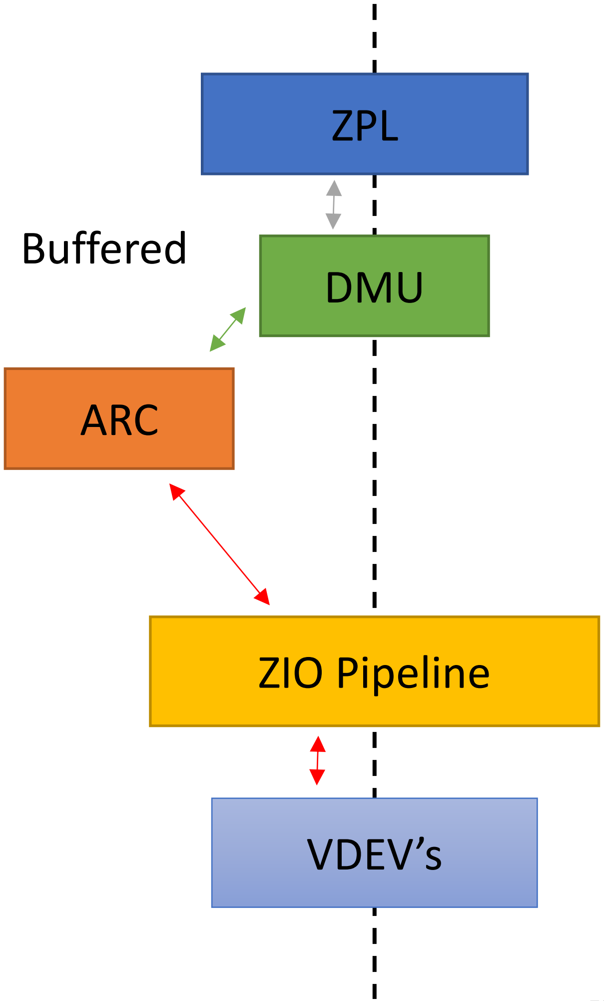




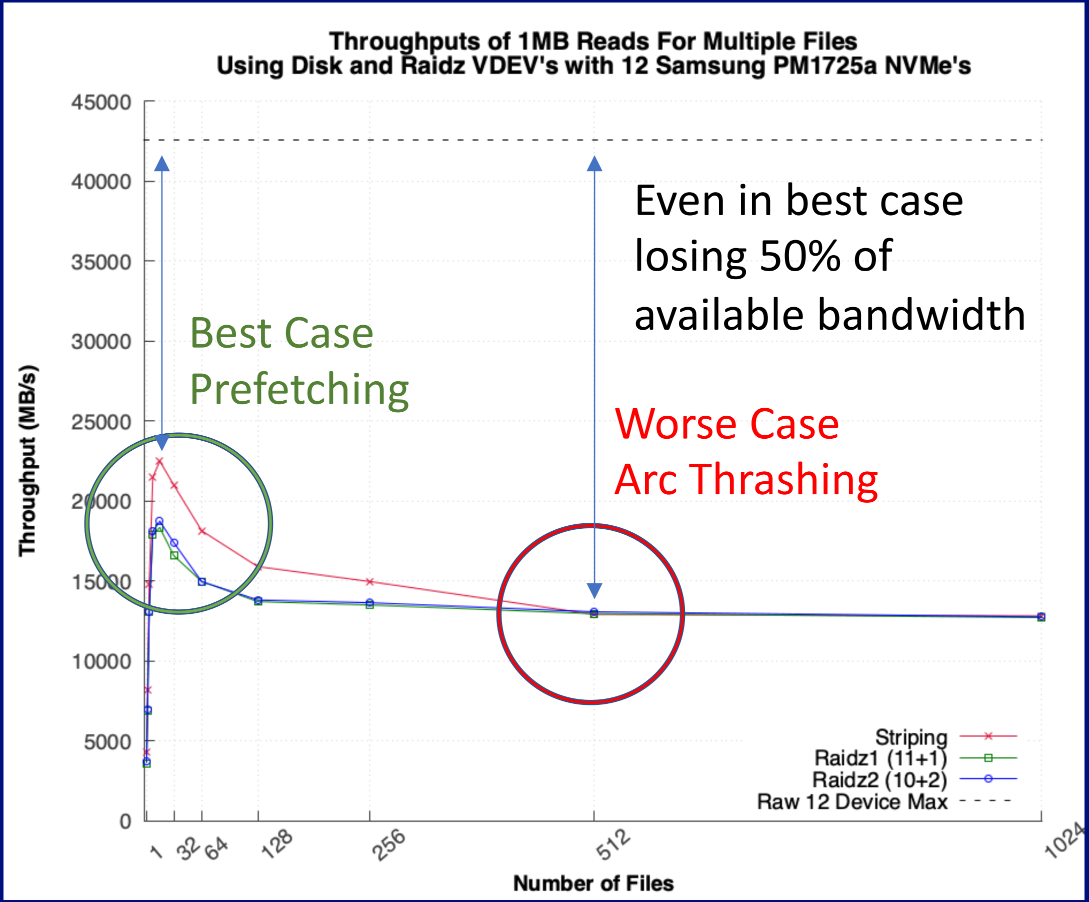
Log Scale

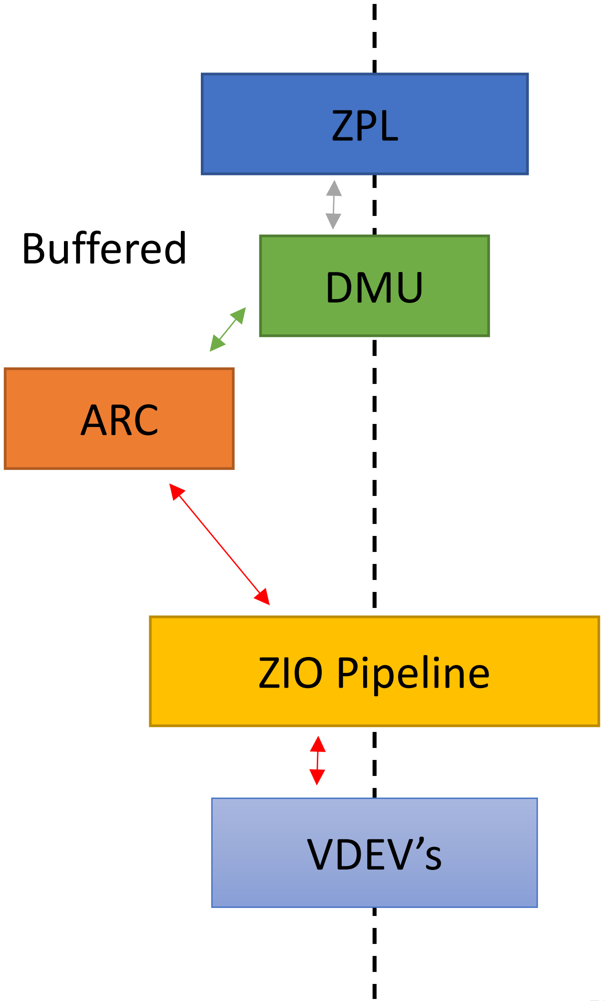
How does ZFS reads data (Highlevel)

- Data is read into and from the
- ARC -> Adjustable Replacement Cache
- Offers memory bandwidth performance (best case)
- Requires read down to disk and copy into the ARC (worse case)

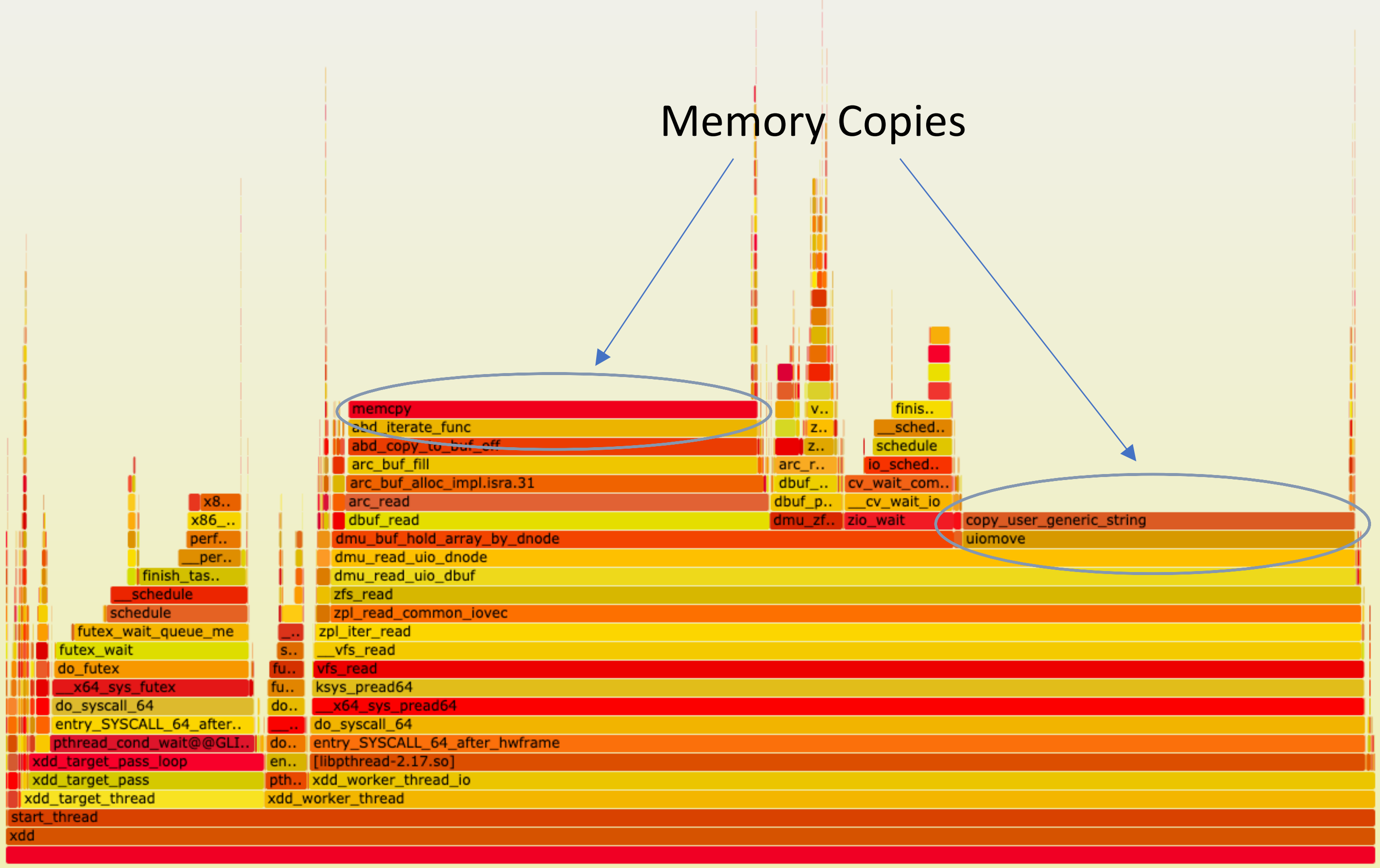


Los Alamos NATIONAL LABORATORY

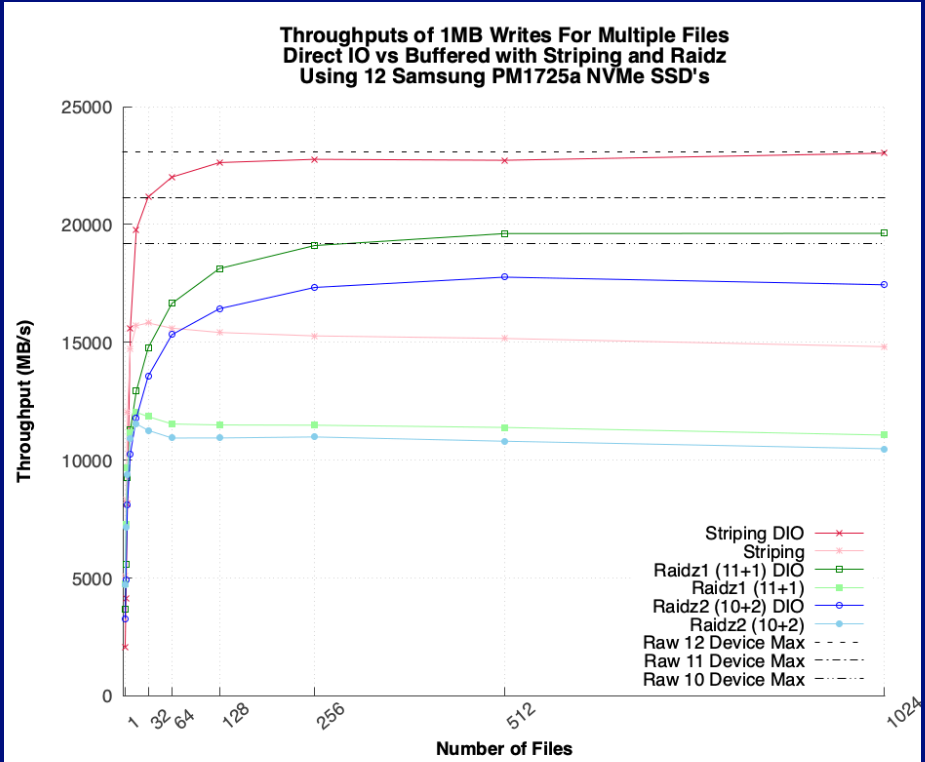

Throughputs of 1MB Reads For Multiple Files
Using Disk and Raidz VDEV's with 12 Samsung PM1725a NVMe's
Throughput (MB/s)
45000
40000
35000
30000
25000
20000
15000
10000
5000
0
1
32
64
128
256
512
1024
Number of Files
Striping
Raidz1 (11+1)
Raidz2 (10+2)
Raw 12 Device Max
Even in best case losing 50% of available bandwidth
Best Case Prefetching
Worse Case Arc Thrashing

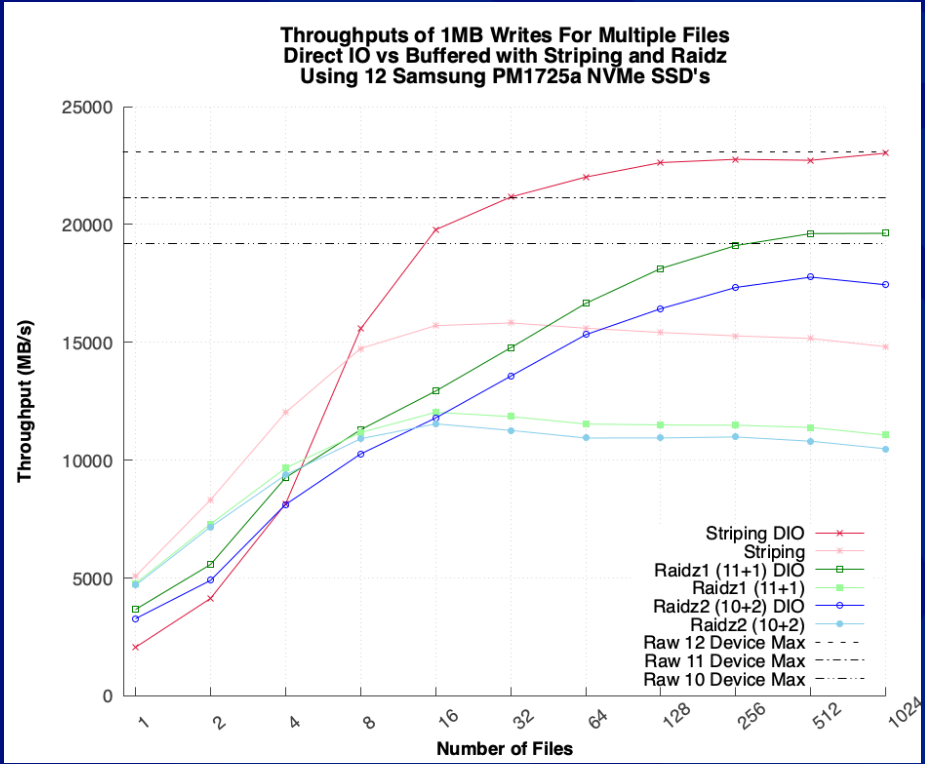

ZPL
Buffered
DMU
ARC
ZIO Pipeline
VDEV's

ZFS Buffered Read Flamegraph



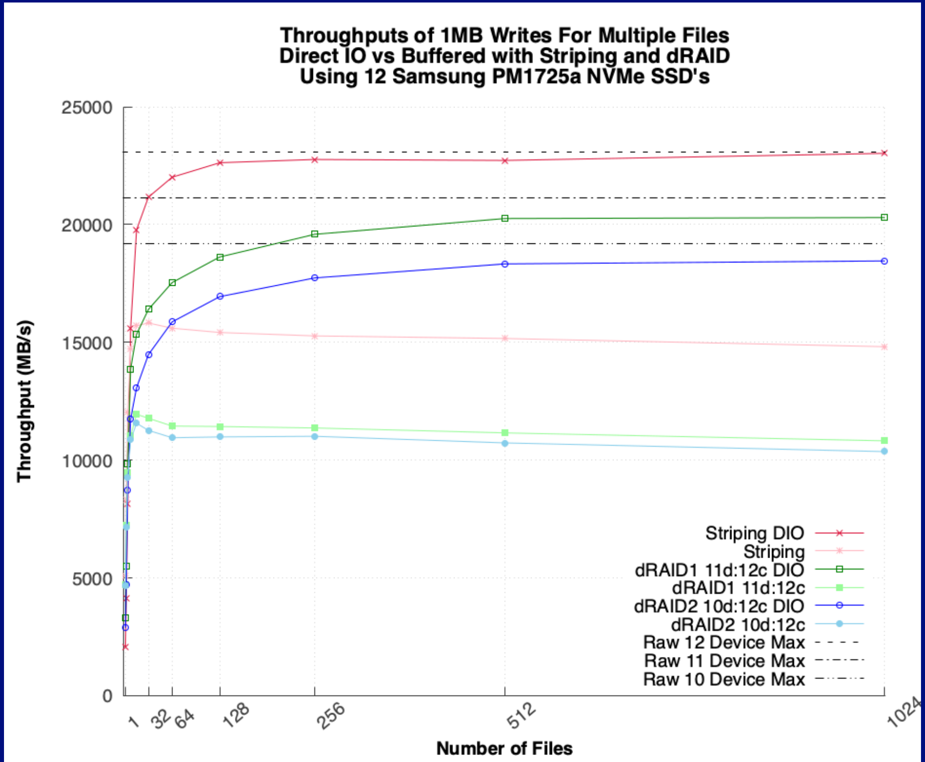
Seq. Write Performance Results: ZFS NVMe Zpools Raidz

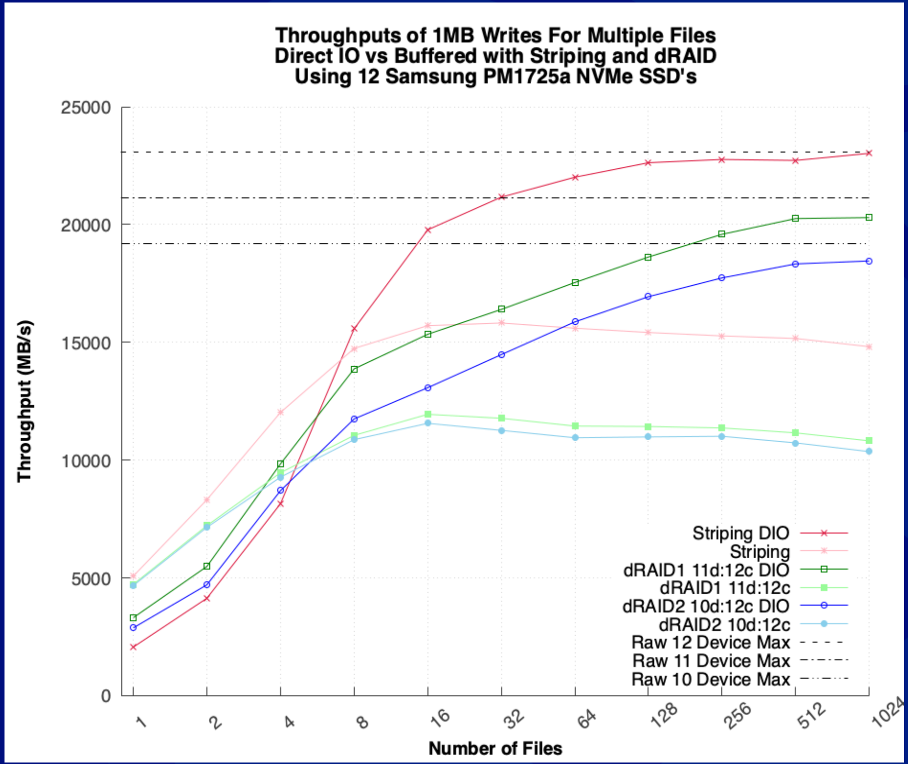




Log Scale

Seq. Write Performance Results: ZFS NVMe Zpools dRAID





Log Scale